**设计文档**

**1 引言**

**1.1 编写目的**

本说明书目的在于明确说明系统各功能的实现方式，指导开发员进行编码。

本说明书的预期读者为：系统设计者、系统开发员。

**1.2 背景**

待开发网页的名称： 游戏论坛

此软件系统任务提出者：客户（游戏爱好者）

此软件系统任务开发者：第七小组

此软件系统任务用户：需要提供用户浏览相关游戏资讯，论坛发表评论，发布帖子等功能。

**2 总体设计**

**2.1 用户相关**

用户调研

|  |  |
| --- | --- |
| 姓名：张小二  性别：男  学历：本科  职业：学生 | 收入：1k  设备：三星笔记本电脑  常用app：支付宝，微信，王者荣耀， |
| 用户特征：  比较关注游戏攻略 ，对价格比较敏感  对新事物接受程度较高，对新游戏有热情  不喜欢复杂的信息 | 用户目标：  希望论坛的功能较全面，既有游戏攻略，同时能提供丰富的游戏资源  希望操作简单，界面整洁 |

项目方向介绍：

游戏论坛网页是服务于游戏爱好者的页面，旨在为用户提供一个了解游戏资讯，发表评论与帖子的交流平台，具有信息共享，资源合理分配，业务协调统一的特点。论坛是玩家和玩家的交流平台：玩家可以通过发帖功能在论坛里互动交友，寻求答案帮助别人，当然偶尔也会成为玩家发泄的平台(这时就需要版主或管理员出面解决)，这使得论坛的社区性变得更加明显，可以增强玩家的互动交流。

论坛也是官方和玩家交流的平台：官方可以在论坛发布重要新闻，可以让玩家及时来了解游戏动向并作出相应的反应。官方还可以在论坛内做游戏的调查，了解玩家的需求。玩家可以发表自己对游戏的建议和意见，由管理员收集整理并给予回复。

**2.2技术选型**

**数据库选型：mysql**

用mysql做中小型项目会比较合适

MySQL的优点：

1.简单易用

MySQL 是一个高性能且相对简单的数据库系统，与一些更大系统的设置和管理相比，其复杂程度较低。

2.价格

MySQL 对多数个人用户来说是免费的。

3.小巧

4.1.1的数据库发行版仅仅只有21M，安装完成也仅仅51M。

4.支持查询语言

MySQL 可以利用 SQL（结构化查询语言），SQL 是一种所有现代数据库系统都选用的语言。也可以利用支持 ODBC（开放式数据库连接）的应用程序，ODBC 是 Microsoft 开发的一种数据库通信协议。

5.开放式的分发

MySQL 容易获得；只要使用 Web 浏览器即可。如果不能理解某样东西是如何起作用的，或者对某个算法感到好奇，可以将其源代码取来，对源代码进行分析。如果不喜欢某些东西，则可以更改它。

从以上查询的资料可以看出，mysql对于中小型web网站的数据库选择有优势，所以数据库选型为mysql。

后端技术选型：Node.js

适合I/O密集型应用

node.js轻量高效，可以认为是数据密集型分布式部署环境下的实时应用系统的完美解决方案。

node.js采用事件驱动、异步编程，为网络服务而设计。

node.js采用非阻塞模式的IO处理，可以使node.js在相对低系统资源耗，拥有出色的负载能力，非常适合用作依赖其它IO资源的中间层服务。

大量Ajax请求的应用

例如个性化应用，每个用户看到的页面都不一样，缓存失效，需要在页面加载的时候发起Ajax请求，NodeJS能响应大量的并发请求。　　总而言之，NodeJS适合运用在高并发、I/O密集、少量业务逻辑的场景。

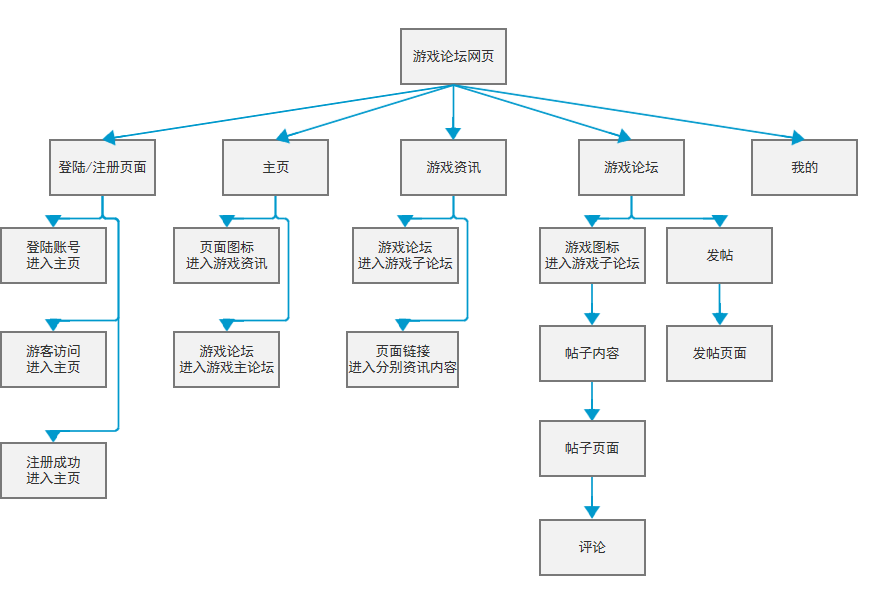
因此后端技术选型可以尝试使用node.js。

还有一个原因，就是学过node.js，使用起来会相对容易一点。

2.3**系统业务结构图**

内容描述：游戏论坛网页是服务于游戏爱好者的页面，旨在为用户提供一个了解游戏资讯，发表评论与帖子的交流平台，具有信息共享，资源合理分配，业务协调统一的特点。它分为登陆/注册面，主页，游戏资讯页面，论坛页面，我的页面等五大主题页面。

 图例：



功能简介：

登陆/注册页面：账号登陆，账号注册，也可以选择游客访问。

主页面： 主页面上有导航栏，可以进入各个主题页面，通过导航栏直接链接到游戏主论坛页面。点击游戏区域的游戏主题可以进入分别的游戏资讯页面，还可以选择二维码加入公众号。

游戏资讯页面：点击游戏论坛可以进入分别的游戏论坛子页面，点击页面中的游戏攻略，赛事资讯等页面内容可以进行游戏各方面资讯的浏览。

游戏论坛页面：选择页面中的不同游戏可以分别进入不同的游戏论坛子页面。

游戏论坛子页面：可以选择发帖进入发帖页面，点击页面中排列的帖子内容进入帖子界面。

发帖界面：编辑好自己想发表的内容，点击提交，提交至论坛子页面。

帖子界面：观看帖子内容和网友评论，页面底部可以发表自己的评论。

我的界面：了解自己的相关信息，如兴趣部落，我的关注，我的帖子等信息。

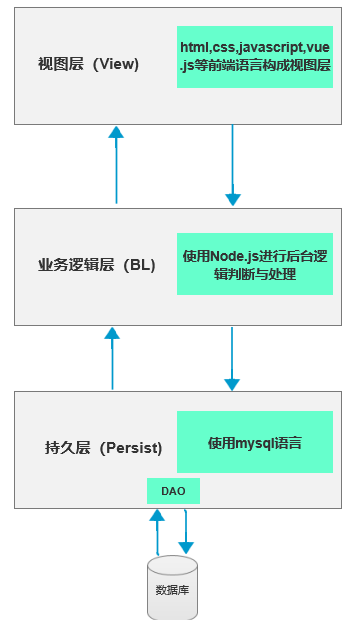
2.4 系统功能层次

运用三层结构进行开发，分别为view(视图层)， BL（业务逻辑层），Persist（持久层）。

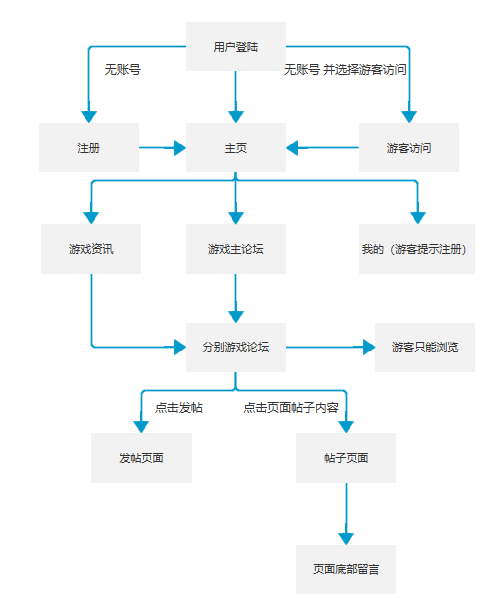
1. 视图层：视图层很好解释 你现在看到的网页 一些界面 都属于表现层的东西可以用一些Html，js,vue.js来实现

(2) 业务逻辑层：业务层用来实现整体的业务逻辑 如 前台获得了数据，逻辑层去解析这些数据，效验这些数据等操作。

(3) 持久层：持久层用来固化数据，如常说的DAO层数据访问层，操作数据库将数据入库。



2.5 功能流程图



功能流程有一大问题点，就是用户作为游客浏览网页和作为已登陆用户进行访问的区别，再进入分别游戏论坛后，用户作为游客时，选择发帖，留言等功能时会自动弹窗提示未登陆需要进行登陆，点击我的页面时也会自动弹窗进行登陆提醒。

2.6数据库设计

一，功能分析

1. 用户注册和登录，后台数据库需要存放用户的注册信息和在线状态信息；
2. 用户发贴，后台数据库需要存放贴子相关信息，如贴子内容、标题等；
3. 浏览、查找帖子，数据库需要实现查看和查找的功能。
4. 论坛版块管理，后台数据库需要存放各个版块信息，如版主、版块名称、贴子数等。

二，实体分析

根据游戏论坛的相关功能，设计的基本实体有User(用户)，Topic(主贴)，Reply(回帖)，Section(板块)。

1.每个实体具有的属性

（1）用户：用户昵称，密码，用户备注，发布记录，关注，用户状态。

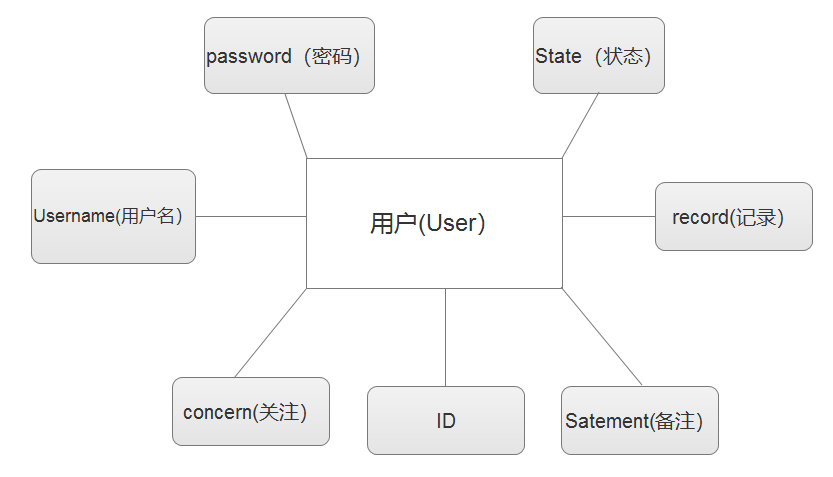
（2）主贴：所属版块，发帖人，标题，正文，发帖时间，回复数。

（3） 回帖：回复主贴ID，所在版块ID，回帖人ID，回复内容，回复时间。

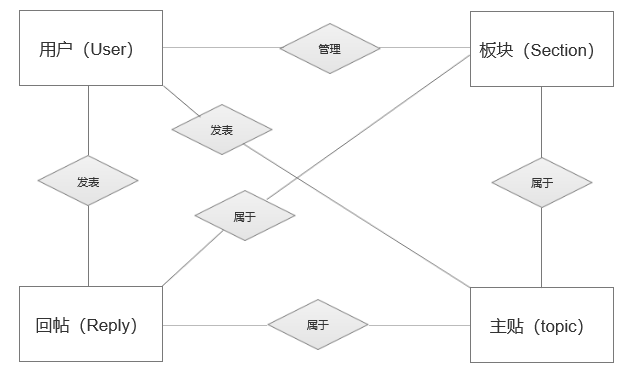
（4）版块：版块ID，版块名称，版块主题，发帖数,版主ID。

**2.E-R图**

**数据库部分的应用分为两块，一是登陆/注册部分和用户信息方面的应用：**



**二是主贴，回帖等论坛功能方面的应用：**



**3.将E-R图转换为表**

将各实体转换为对应的表，将各属性转换为对应表的列。

在表之间建立主外键，表示实体之间的映射关系。

User(用户表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 能否为空 | 备注 |
| UID | 用户ID | Int | 否 | 主键 自定增长 |
| Uname | 用户昵称 | Varchar | 否 | 唯一值 |
| Upassword | 密码 | Varchar | 否 |  |
| Ustate | 用户状态 | Int |  |  |
| Urecord | 发帖记录 | Varchar |  |  |
| Uconcern | 用户关注 | Varchar |  |  |
|  |  |  |  |  |

Section(版块表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 能否为空 | 备注 |
| SID | 版块ID | Int | 否 | 主键，自定增长 |
| SName | 板块名称 | Varchar | 否 | 唯一值 |
| SMasterID | 版主ID | Varchar | 否 | 外键（引用用户表的UID） |
| STopicCount | 发帖数 | Int |  |  |
| SProfile | 版块主题 | Varchar |  |  |

STopic(主贴表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 能否为空 | 备注 |
| TID | 发帖表ID | Int | 否 | 主键自定增长 |
| TSID | 所在版块ID | Int | 否 | 外键（引用版块表的SID） |
| TUID | 发帖人 | Int | 否 | 外键（引用用户表的UID） |
| TTopic | 标题 | varchar | 否 |  |
| TContents | 正文 | varchar | 否 |  |
| TTime | 发帖时间 | Datatime | 否 |  |
| TReplyCount | 回复数 | int |  |  |

SReply(回帖表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 能否为空 | 备注 |
| RID | 回帖表ID | Int | 否 | 自定增长 |
| RTID | 回复主贴ID | Int | 否 | 外键（引用主帖表的TID） |
| RSID | 所在版块ID | Int | 否 | 外键（引用版块表的SID） |
| RUID | 回帖人ID | Int | 否 | 外键（引用用户表的UID） |
| RContent | 回帖内容 | Varchar | 否 |  |
| RTime | 回帖时间 | Datetime | 否 |  |

2.7 接口设计

接口设计采用用户代号U001进行分别设计。设计针对发帖与回帖等消息体变动的部分。

1. 发帖部分

增加帖子

|  |  |
| --- | --- |
| URI | /Topics |
| 动词 | post |
| 请求消息体 | { “name”:”U001”,  “title”:”Game”,  “time”:”11:11”,  “content”:”xxxxx”  } |
| 返回消息体 | {“id”:”001”,  “title”:”Game”,  “name”:”Game”,  “time”:”11:11”,  “content”:”xxxxx”  } |

查询帖子

|  |  |
| --- | --- |
| URI | /Topic?name=”Game” |
| 动词 | get |
| 请求消息体 |  |
| 返回消息体 | [{  “name”:”U001”,  “title”:”Game”,  “time”:”11:11”,  “content”:”xxxxx”}  {…..}  ] |

删除帖子

|  |  |
| --- | --- |
| URI | /topic/U001 |
| 动词 | delete |
| 请求消息体 |  |
| 返回消息体 |  |

1. 回帖部分

增加回帖

|  |  |
| --- | --- |
| URI | /reply |
| 动词 | post |
| 请求消息体 | { “name”:”U001”,  “time”:”11:11”,  “content”:”xxxxx”  } |
| 返回消息体 | { id=”001”  “name”:”U001”,  “time”:”11:11”,  “content”:”xxxxx”  } |

删除回帖

|  |  |
| --- | --- |
| URI | /reply/U001 |
| 动词 | delete |
| 请求消息体 |  |
| 返回消息体 |  |